

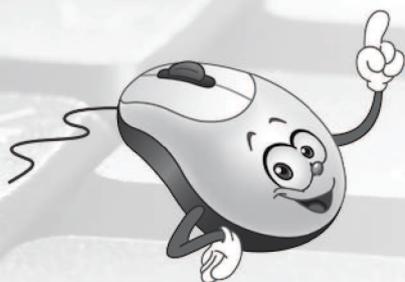
גלית ברעם

עיצוב תוכנה בשפת #C

מדעי המחשב ב'

2 יחידות לימוד

סמל שאלון 899205



רכס

פרויקטים חינוכיים בע"מ

אובייקטים
#C עיצוב תוכנה בשפת

גלית ברעם

© 2014 כל הזכויות שמורות
לרכס פרויקטים חינוכיים בע"מ ולמחברת
Printed in Israel 2014

אין לשכפל, להעתיק, לצלם, להקליט, לתרגם, לאחסן במאגר מידע, לשדר או לקלוט בכל דרך או אמצעי אלקטרוני, אופטי או מכני או אחר, כל חלק שהוא מספר זה. שימוש מסחרי, מכל סוג שהוא, בחומר הכלול בספר זה אסור בהחלט אלא ברשות מפורשת בכתב מן המו"ל.

רכס פרויקטים חינוכיים בע"מ
ת"ד 75 אבן יהודה 40500
טלפון 073-2550000, פקסימיליה 073-2550055
כתובתנו באינטרנט: www.reches.co.il
E-mail: main@reches.co.il

עשינו כמיטב יכולתנו לאתר את בעלי הזכויות של כל החומר ממקורות חיצוניים. אנו מתנצלים על כל השמטה או טעות. אם יובא הדבר לידיעתנו נפעל לתקנו במהירות הבאות.

מסת"ב 978-965-558-064-8 ISBN

1 2 3 4 5 6 7 8 9 10

תוכן עניינים

חלק א' – הגדרות וממשקים

7	אובייקטים
11	רשימה
13	מחסנית
15	תור
17	עץ בינרי

חלק ב' – מבחנים במתכונת בחינת הבגרות

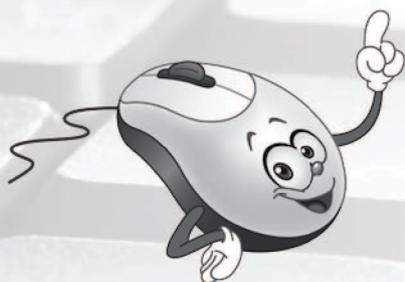
21	מבחן מספר 1
24	מבחן מספר 2
26	מבחן מספר 3
28	מבחן מספר 4
30	מבחן מספר 5
34	מבחן מספר 6
37	מבחן מספר 7
41	מבחן מספר 8
45	מבחן מספר 9
49	מבחן מספר 10

תשובות

53	תשובות מבחן מספר 1
58	תשובות מבחן מספר 2
62	תשובות מבחן מספר 3
66	תשובות מבחן מספר 4
70	תשובות מבחן מספר 5
73	תשובות מבחן מספר 6
76	תשובות מבחן מספר 7
82	תשובות מבחן מספר 8
86	תשובות מבחן מספר 9
90	תשובות מבחן מספר 10

חלק א'

עיצוב תוכנה - הגדרות וממשקים



אובייקטים

הגדרת אובייקטים

אובייקט הוא עצם המציין חי, צומח או דומם. קיימים סוגים שונים של עצמים ושניים מהם מפורטים להלן.

עצם מוחשי הוא מילה הנדפסת באחד החושים. לדוגמה: ספר, בן אדם, שולחן, שמש, עט וכדומה. עצם מופשט הוא מילה אשר אינה נדפסת באחד החושים. לדוגמה: מקצוע, ציון, בית ספר, מספר חשבון בנק וכדומה.

כאשר אנו מתבוננים בעצם מכל סוג שהוא, אנו יכולים לתאר אותו בתכונות מסוימות, או לציין את הפעולות שאפשר לבצע בו.

לדוגמה:

העצם: מלבן.

התכונות המתארות את העצם מלבן

1. מידת אורך.

2. מידת רוחב.

הפעולות שאפשר לבצע בעצם מלבן

1. חישוב שטח.

2. חישוב היקף.

כמובן, אפשר גם להרחיב את מספר התכונות המגדירות מלבן, ולתאר עצם "תיבה צבעונית".

1. מידת אורך.

2. מידת רוחב.

3. מידת גובה.

4. צבע.

הפעולות שאפשר לבצע בעצם "תיבה צבעונית"

1. חישוב שטח.

2. חישוב היקף.

3. חישוב נפח.

4. החלפת צבע.

- כאשר רוצים לייצג את העצם באופן ממוחשב, בשפת #C, יש לייצג את העצם באמצעות מחלקה class. בניית מחלקה לעצם נעשית לפי הכללים הבאים:
1. להגדיר שם למחלקה (לפי כללי הגדרת שם משתנה).
 2. להגדיר לכל תכונה את סוג הנתון שלה.
לדוגמה:
מידת אורך היא מספר ממשי (double).
 3. להגדיר הגדרה פרטית או ציבורית.
הגדרה פרטית, כלומר private, כאשר רק המחלקה עצמה מכירה את הנתונים.
הגדרה ציבורית, כלומר public, כאשר כל המשתתפים בתוכנית מכירים את התכונות.
את התכונות מקובל להגדיר כ-private, ואילו את הפעולות מקובל להגדיר כ-public.
 4. לאחר הגדרת התכונות יש להגדיר פעולה בונה למחלקה, באותו שם של המחלקה.
תפקידה של הפעולה הבונה היא להגדיר את העצם, ולהקצות לו מקום בזיכרון. כמו כן הפעולה הבונה כוללת הגדרה ראשונית ואתחול הנתונים בפעם הראשונה.
את הפעולה הבונה מריצים פעם אחת בתחילת התוכנית, כאשר מתחילים להשתמש בעצם.
 5. להגדיר פעולת Get.
פעולה זו נבנית לכל אחת מהתכונות, ומטרתה לאחזר את הנתון מתוך המחלקה. כלומר, להחזיר את הנתון מתוך העצם.
מכיוון שבמחלקה יש לכל תכונה מספר תכונות, יש פעולת Get לכל תכונה.
לדוגמה:
מידת רוחב מיוצגת בתכונה width.
לכן פעולת ה-Get שלה תהיה Getwidth().
פעולה זו תחזיר את הנתון אשר נמצא בתכונה width.
 6. להגדיר פעולה Set שלה תהיה Setwidth(double X).
פעולה זו נבנית לכל אחת מהתכונות, ומטרתה לעדכן את הנתון מתוך המחלקה.
מכיוון שבמחלקה יש לכל תכונה מספר תכונות, יש פעולת Set עם הנתון שהיא מכילה.
לדוגמה:
מידת רוחב מיוצגת בתכונה width.
לכן פעולת ה-Set שלה תהיה double.
פעולה זו תעדכן את הנתון אשר נמצא בתכונה width, לערכו החדש X.
 7. להגדיר פעולות נוספות על פי הדרישות וההגדרות של המתכנת.
לדוגמה:
א. חישוב שטח.
ב. חישוב היקף.

```
public class rectangle
{
    private double length; //מספר ממשי/ אורך מלבן מסוג מספר ממשי/
    private double width; //מספר ממשי/ רוחב מלבן מסוג מספר ממשי/

    public rectangle (double x, double y) //הפעולה בונה בשם זהה לשם המחלקה
    //פעולה בונה אשר מגדירה את העצם "מלבן", ומאתחלת את הנתונים
    {
        this.length = x; //מאתחל את הנתון אורך בעצם מלבן/
        this.width = y; //מאתחל את הנתון רוחב בעצם מלבן/
    }

    public double Getlength() // פעולה המאחזרת את הנתון של אורך מלבן
    {
        return(this.length);
    }

    public double Getwidth ()//פעולה המאחזרת את הנתון של רוחב מלבן/
    {
        return(this.width);
    }

    public void Setlength (double newX)
    // פעולה המעדכנת את הערך של אורך מלבן לערך חדש newX
    {
        this.length = newX;
    }

    public void Setwidth (double newY)
    // פעולה המעדכנת את הערך של רוחב מלבן לערך חדש newY
    {
        this.width = newY;
    }
}
```

```
public double area ()//פעולה המחשבת שטח מלבן
{
    return(this.length*this.width);
}
```

```
public double perimeter ()//פעולה המחשבת היקף מלבן
{
    return(2*this.length+2*this.width);
}
```

פנייה לתכונה הנמצאת בתוך עצם (אובייקט)

בפנייה לתכונה הנמצאת באובייקט (עצם) קיימות שתי אפשרויות, המפורטות להלן:

אפשרות ראשונה

אם הפעולה נמצאת במחלקת האובייקט, השימוש בתכונה נעשה בפעולה עם `this`. לדוגמה: `this.length`. פעולה זו נקראת פעולה פנימית, מכיוון שהיא נמצאת בתוך המחלקה (`class`).

אפשרות שנייה

אם הפעולה נמצאת מחוץ למחלקת האובייקט, השימוש בתכונה נעשה בפעולה `.Get`. לדוגמה: `Getlength()`. פעולה זו נקראת פעולה חיצונית, מכיוון שהיא נמצאת מחוץ למחלקה (`class`).

רשימה

עד כה שמרנו את הנתונים במערך.

מערך הוא:

- שמור בזיכרון באופן רציף.
החיסרון בהגדרת מקום רציף בזיכרון הוא חיפוש מקום רציף של כל המערך. אם נוצר מצב שבו אין מקום סדרתי בזיכרון ואין מקום פנוי, המחשב לא יכול להגדיר את המקום, ולפיכך התוכנית לא תוכל לרוץ.
- כמות האיברים במערך מוגדרת מראש (בתחילת התוכנית). אין אפשרות לשנות את גודל המערך בזמן הרצת התוכנית.
החיסרון בהגדרה מראש של גודל המערך: ייתכן שיווצר מצב שבו גודל המערך שהוגדר גדול מדי, ולפיכך יש תאים ריקים רבים (המבזבזים את זיכרון המחשב). במקרה כזה גם יכול להיווצר מצב שהוגדר מעט מדי מקום, ובזמן הרצת התוכנית לא נוכל לשמור את עודף הנתונים.
לנוכח שני החסרונות הבולטים שהוצגו לעיל, השימוש במערך הוא מוגבל. עקב כך יש צורך למצוא דרך אחרת לשמירת הנתונים.

ברשימה

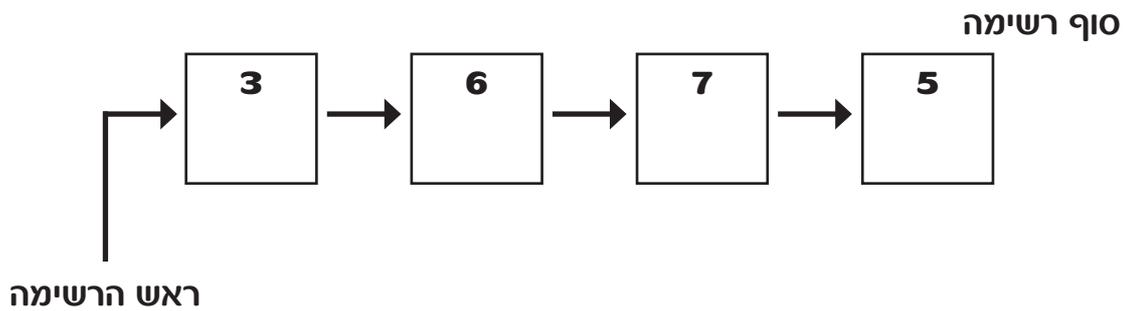
הגדרה: אוסף של איברים מסוג מסויים.

כל איבר רשימה מוגדר משני חלקים:

1. ערך הנתון של האיבר - DATA.
2. ערך הכתובת של האיבר הבא ברשימה.

נתון	כתובת

לפי אופן בניית איבר ברשימה, שמירת הנתונים בזיכרון אינה רציפה. הסיבה לכך היא: מכיוון שרוצים להוסיף איבר לרשימה, "מחפשים" מקום בזיכרון לאיבר אחד, ומחברים את הכתובת. כמו כן אין הגדרה בתחילת התוכנית של מספר האיברים ברשימה, וכך אפשר להוסיף בזמן ריצת התוכנית איברים נוספים.



המספרים בתוך הם המספרים ברשימה, והם מאפיינים את הכתובת של האיבר הבא בזיכרון.

כללים לשימוש ברשימה

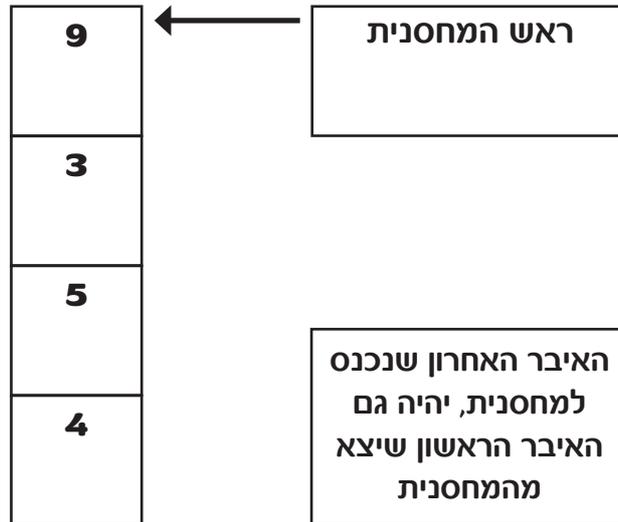
- הרשימה נשמרת בזיכרון באופן לא רציף.
- כמות הנתונים שאפשר להוסיף ברשימה היא בלתי מוגבלת (הגבלה עד גודל זיכרון המחשב).
- המעבר על הרשימה נעשה באופן סדרתי, כדי להגיע לאיבר במקום ה-I יש לעבור על I-1 נתונים.
- לכל רשימה יש איבר ראשון ברשימה שבו נמצאת הכתובת של הנתון הראשון ברשימה, והוא נקרא `GetFirst()`.
- סיום רשימה מסומן ב-`null`.
- הוספת נתון לרשימה מתאפשרת בכל מקום שנבחר.

ממשק לרשימה

<code>Public List()</code>	פועלה בונה רשימה ריקה, ומאתחלת אותה
<code>Public Node <type > GetFirst()</code>	הפעולה המחזיקה את הכתובת של האיבר הראשון ברשימה
<code>Public Node <type > Insert (Node <type > p, x)</code>	הפעולה מקבלת משתנה X ומקום P, ומכניסה את ערך של X לרשימה במקום P
<code>Public Node <type > Remove (Node <type > p)</code>	הפעולה מוציאה את האיבר במקום ה-P, ומחזירה את ערכו
<code>Public bool IsEmpty()</code>	הפעולה מחזירה "אמת", אם הרשימה ריקה, אחרת מחזירה "שקר"
<code>Public string ToString()</code>	הפעולה מחזיקה מחרוזת עם כל איברי הרשימה

מחסנית

הגדרה: אוסף של נתונים מסוג מסויים בעלי אופן של סדר הראשון שנכנס הוא הראשון שיוצא.



כללים לשימוש במחסנית

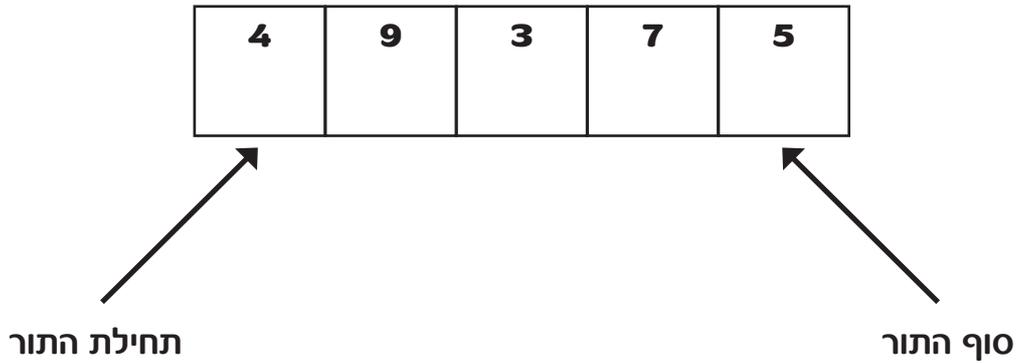
- מחסנית נשמרת בזיכרון באופן רציף.
- למחסנית מוגדר סדר בנתונים.
- האיבר האחרון שנכנס הוא גם האיבר הראשון שיוצא.
- כמות הנתונים שאפשר להוסיף במחסנית היא בלתי מוגבלת (הגבלה עד גודל זיכרון המחשב).
- המעבר על המחסנית הוא באופן סדרתי, כדי להגיע לאיבר במקום ה-I, יש לעבור על I-1 נתונים.
- לכל מחסנית יש ראש מחסנית, שהוא הנתון האחרון שנכנס למחסנית.
- סיום מעבר על המחסנית נעשה כאשר המחסנית ריקה.
- כאשר מוציאים נתון מהמחסנית, הוא אינו נשמר ונמחק מהמחסנית. לכן אם ברצוננו לשמור על הנתונים שאנו מוציאים מהמחסנית, עלינו להשתמש במחסנית עזר.
- הוספת נתון למחסנית אפשרית רק בראש המחסנית.

ממשק למחסנית

Public stack()	פועלה בונה מחסנית ריקה, ומאתחלת אותה
Public void Push(<type > x)	הפעולה מקבלת משתנה X, ודוחפת אותו לראש המחסנית
Public < type> Pop ()	הפעולה מוציאה את האיבר הראשון מראש המחסנית ומחזירה אותו (בסיום הפעולה ראש המחסנית <u>משתנה</u>)
Public < type> Top	הפעולה מחזירה את הערך בראש המחסנית, בלי שהערך ייצא מראש המחסנית (ראש המחסנית <u>לא</u> משתנה)
Public bool IsEmpty()	הפעולה מחזירה "אמת", אם המחסנית ריקה, אחרת מחזירה "שקר"
Public string ToString()	הפעולה מחזיקה מחרוזת עם כל איברי המחסנית

תור

הגדרה: אוסף של נתונים בעל אופי של סדר מסויים הראשון שנכנס הוא ראשון שיוצא.



כללים לשימוש בתור

- לתור מוגדר סדר בנתונים.
- האיבר האחרון שנכנס הוא האיבר האחרון שיוצא.
- כמות הנתונים שאפשר להוסיף בתור היא בלתי מוגבלת (הגבלה עד גודל זיכרון המחשב).
- המעבר על התור הוא באופן סדרתי, כדי להגיע לאיבר במקום ה-I, יש לעבור על I-1 נתונים.
- לכל תור יש ראש תור, שהוא הנתון הראשון שנכנס לתור.
- לכל תור יש סוף תור, שהוא הנתון האחרון שנכנס לתור.
- סיום מעבר על התור נעשה כאשר התור ריק.
- כאשר מוציאים נתון מתור הוא אינו נשמר ונמחק מהתור. לכן אם ברצוננו לשמור על הנתונים שאנו מוציאים מתור, עלינו להשתמש בתור עזר.
- הוספת נתון לתור אפשרית רק בסוף התור.

ממשק לתור

Public Queue()	פועלה בונה תור ריקה, ומאתחלת אותה
Public void Insert (<type > x)	הפעולה מקבלת משתנה X, ומכניסה את ערך של X לתור
Public < type> Head ()	הפעולה מחזירה את האיבר הראשון מראש התור ומחזירה אותו (ראש תור לא משתנה)
Public < type> Remove()	הפעולה מוציאה את הערך מראש התור, בלי שהערך ייצא מראש התור (ראש התור לא משתנה)
Public bool IsEmpty()	הפעולה מחזירה "אמת", אם התור ריק, אחרת מחזירה "שקר"
Public string ToString()	הפעולה מחזיקה מחרוזת עם כל איברי התור